LINKgroup

Distance Learning System

Rad sa petljama

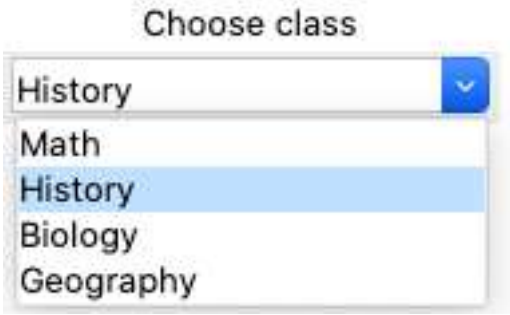
Python and programming fundamentals

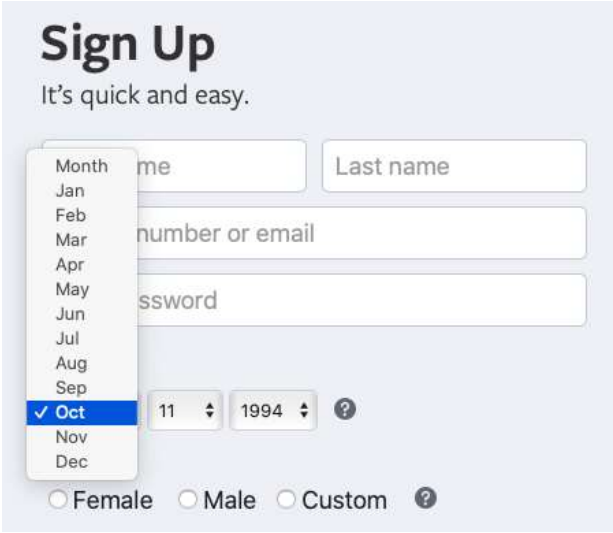
Petlje u programiranju

- U programiranju, petlja nam omogućava da određeni kod izvršimo više puta
- Tokom izvršavanja, postoji mogućnost modifikacije vrednosti promenljivih
- Trajanje petlje se kontroliše nekom kontrolnom promenljivom ili iteratorom

Gde srećemo petlje

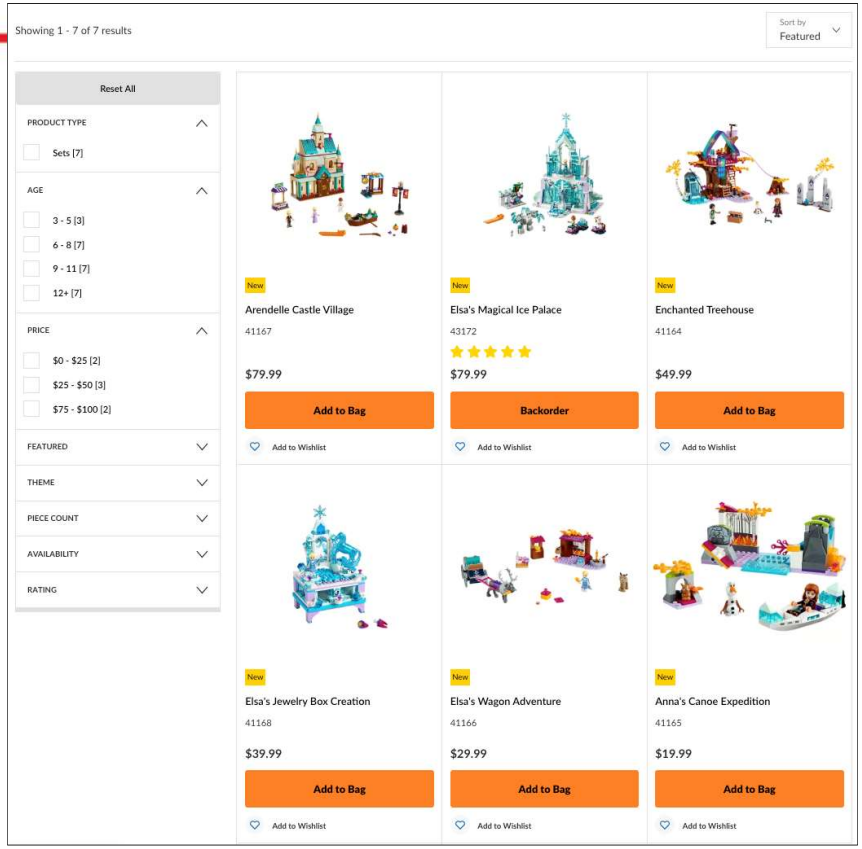
- U **padajućim menijima**, svaka stavka se iscrtava istom tehnikom, ali se vrednost





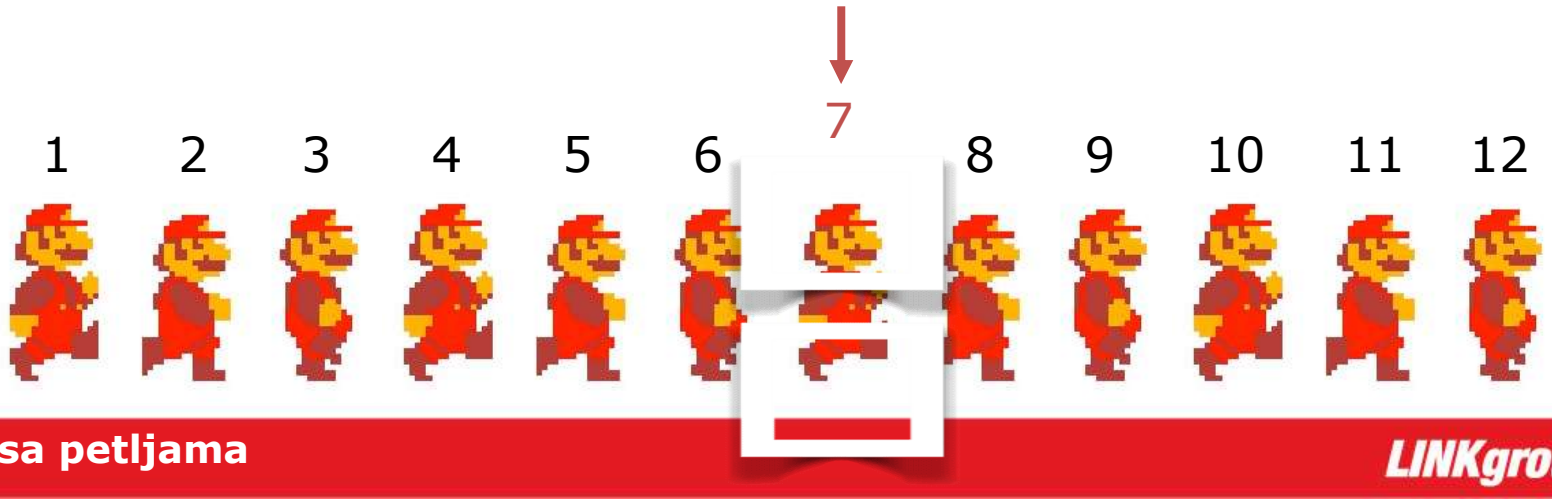
Gde srećemo petlje

- U web prodavnici, gde god postoji listanje proizvoda ono je realizovano pomoću petlje



Gde srećemo petlje

- Čak ni najjednostavnija igra ne može funkcionisati bez petlje



Gde srećemo petlje

- Kompleksnije igre, sa animacijom, sadrže mnoštvo petlji

Vrste petlji u Python-u

- Python poznaje dve vrste petlji

for

- Primenjuje se nad kolekcijama
- Zna se broj iteracija
- Nije potrebno kontrolisati je ručno

while

- Ne zahteva kolekciju
- Ne zna se broj iteracija
- Mora se kontrolisati “ručno”
- Veća opasnost od

for petlja

```
for i in [1,2,3]:
    print(i)
```

- Primenjuje se nad kolekcijama
- Zna se broj iteracija
- Nije potrebno kontrolisati je ručno
- Manja je opasnost od zaključavanja (mrtve petlje)

for petlja

- For petlja zahteva kolekciju podataka da bi mogla da se izvrši
- Ova kolekcija može već postojati u programu, ali je takođe možemo generisati samo u svrhu izvođenja petlje

aktuelni član kolekcije

generator kolekcije

```
for i in range(10):
    print(i)
```

telo - kod koji se ponavlja

```
0
1
2
3
4
5
6
7
8
9
```

Vežba 1 (ppf-ex06 years.py)

- U programu su definisane promenljive startDate i endDate koje predstavljaju početnu i krajnju godinu
- Potrebno je kreirati program koji
- Aplikacija treba da
  - Aplikacija treba da ima izlaz slican sledecem

```
********* Allowed years *********
2010
2011
2012
2013
2014
2015
*********************************
```

Napomena:
Komanda range() može se koristiti sa parametrima na primer:

```
range(1,10)
```

Generiše kolekciju od broja 1 do broja 9

Vežba 2 (ppf-ex06 multable.py)

- Korisnik unosi broj
- Potrebno je kreirati tablicu množenja do 3 za brojeve od 1 do vrednosti unetog broja
- Na primer, ako korisnik unese broj 5, tablica bi trebala biti kao na slici

```
Enter number: 5
1       2       3
**********************
1       2       3
2       4       6
3       6       9
4       8       12
5       10      15
```

Napomena:
Da sprečimo štampanje novog reda u komandi print, koristimo parametar end:
`print('Number is:',5,end='')`
Za tab oznaku unutar štampanog teksta, koristimo kod **\t**

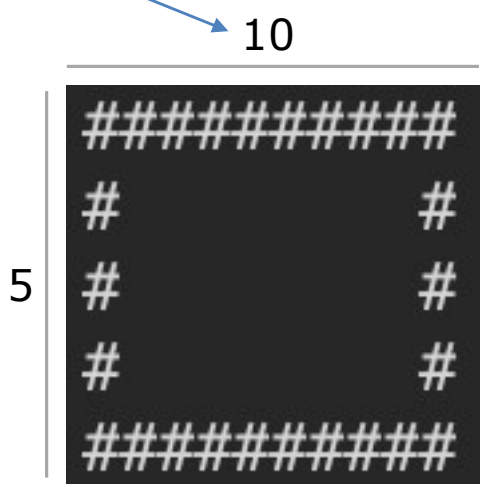
Vežba 3 (ppf-ex06 rect.py)

- U programu se nalaze dve promenljive: **h** i **w**
- Potrebno je, pomoću jedne petl

  iscrtati pravougaonik

  širine **w** i visine **h**.

```
w = 10
h = 5
```



Ugnježđena for petlja (petlja u petlji)

- Unutar jedne petlje se može naći neograničena količina koda, koja može takođe sadržati jednu ili više for petlji
- U sledećem primeru, spoljašnja petlja se izvršava 10 puta, a unutrašnja takođe 10 puta. Ovo znači da će se telo unutrašnje petlje izvršiti 100 puta

```
for i in range(10):
    for j in range(10):
        print("Hello")
```

Ugnježđena for petlja (petlja u petlji)

- Prilikom izvršavanja petlje u petlji, treba voditi računa o izloženim promenljivima

Ispravno

```
for i in range(10):
    for j in range(10):
        print("Hello")
```

Može biti problem

```
for i in range(10):
    for i in range(10):
        print("Hello")
```

Vežba 4 (ppf-ex06 triangle.py)

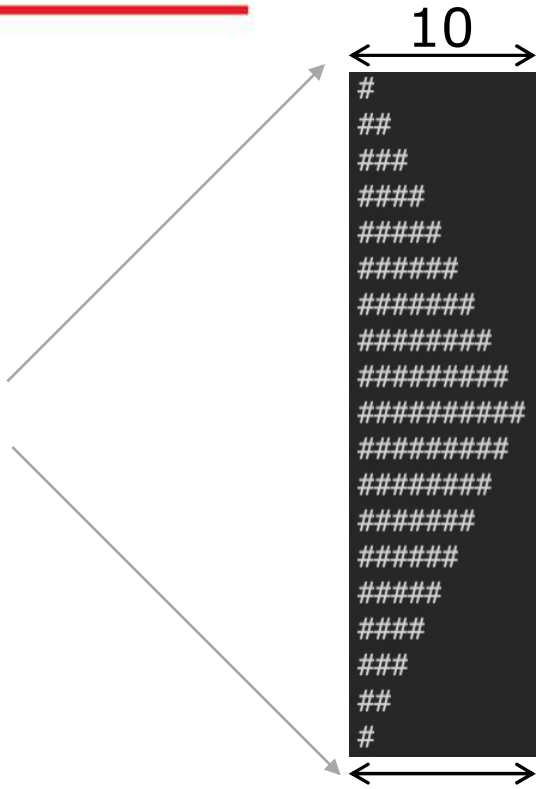
- U aplikaciji postoji promenljiva H
- Potrebno je, pomoću petlje, kreirati trougao kao na

```
H = 10
```

Vežba 5 (ppf-ex06 triangle1.py)

- U aplikaciji postoji promenljiva H
- Potrebno je, pomoću petlje, kreirati trougao kao na slici

```
H = 10
```



Vežba 6 (ppf-ex06 anim.py)

- U programu je definisana promenljiva **target**
- Potrebno je animirati oznaku # tako da se kreće s leva na desn[redacted] # [redacted]sane promenljivom target

Napomena:

Moguće je vratiti se na početak reda u konzoli, oznakom \r

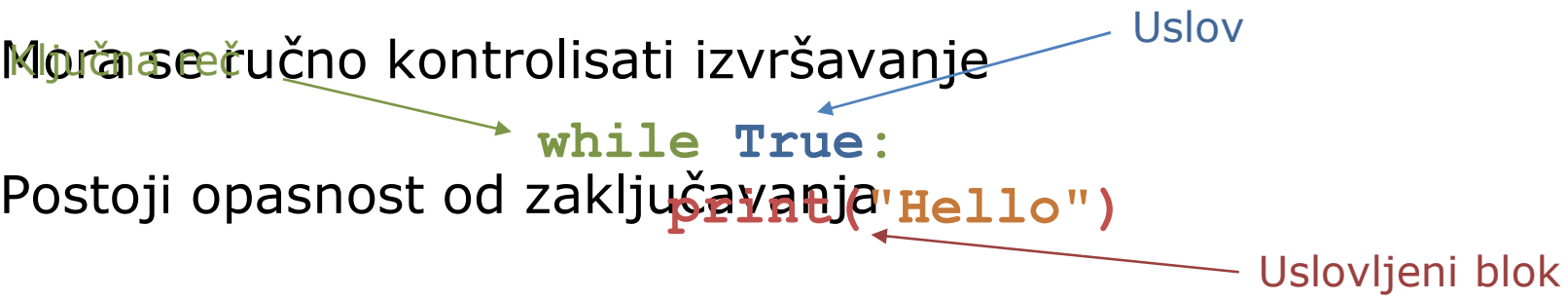
```
print("\r",end="")
```

Moguće je pauzirati izvršavanje programa dodavanjem sledećih linija

```
import time #na početku fajla
…
time.sleep(0.1) #tamo gde je potrebno pauzirati. 0.1 = 100 milisekundi
```

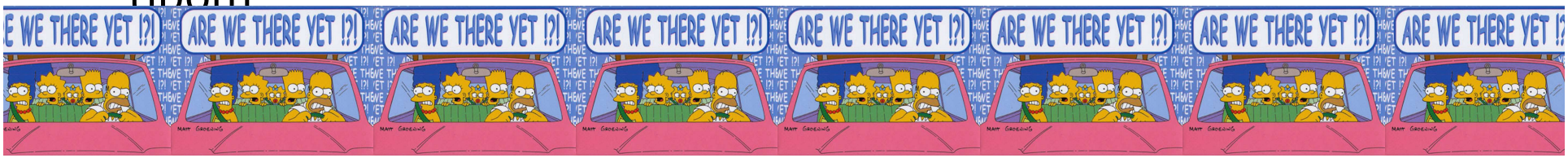
while petlja

- While petlja se izvršava neodređeno puta
- Ova petlja je najbliža implementacija standardne petlje u Python-u
- Mora se ručno kontrolisati izvršavanje
- Postoji opasnost od zaključavanja



while petlja

- Uslov u while petlji je standardan izraz koji rezultuje boolean tipom



Poruka Hello se ispisuje beskonačno

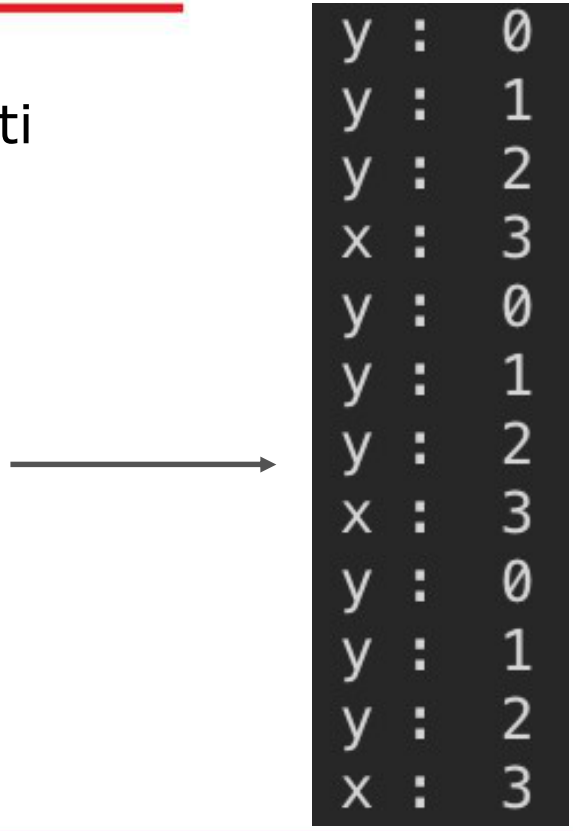
```
x = 1
while x == 1:
    print("Are we there yet")
```

```
while(True):
    print("Are we there yet")
```

Ugnježđena while petlja

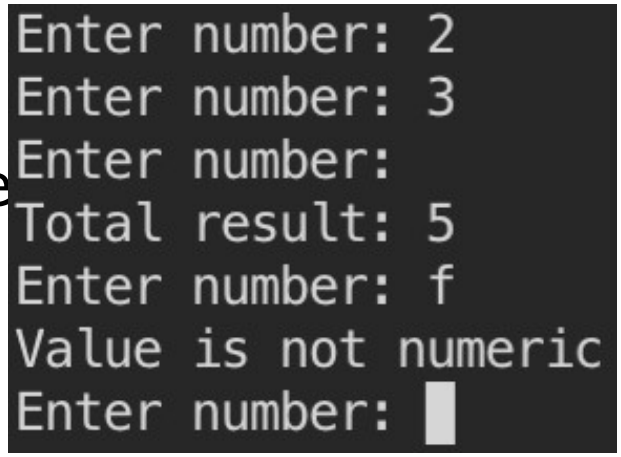
- Petlja while se, kao i for petlja, može gnjezditi

```
x = 0

while x < 3:

    y = 0

    while y < 3:

        print("y : ", y)
```



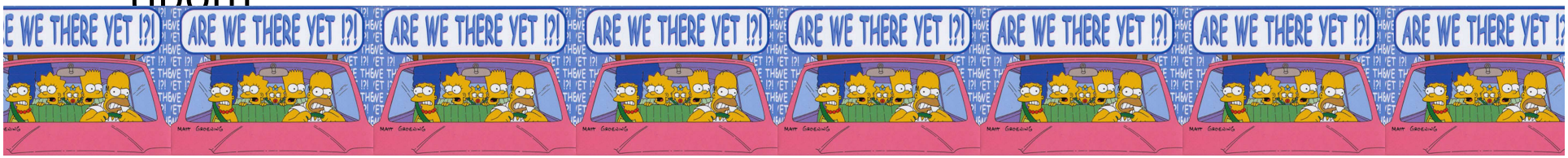
Vežba 7 (ppf-ex06 calc.py)

- Program traži od korisnika unos broja
- Sve dok korisnik unosi numeričke vrednosti, program ih sabira
- Kada korisnik pritisne taster enter ume vrednosti program ispisuje zbir i počinje ponovno sabiranje



while petlja

- Uslov u while petlji je standardan izraz koji rezultuje boolean tipom



Poruka Hello se ispisuje beskonačno

```
x = 1
while x == 1:
    print("Are we there yet")
```

```
while(True):
    print("Are we there yet")
```

Kontrola petlje

- For i while petlje imaju mogućnost kontrole na nivou iteracije

  continue - momentalno prekida trenutnu iteraciju i prelazi na sledeću

  break - momentalno prekida petlju

Beskonačno se ispisuje Hello ali nikad World

```
while True:
    print("Hello")
    continue
    print("World")
```

Samo jednom se ispisuje Hello

```
while True:
    print("Hello")
    break
```

Slučaj kompletiranja petlje

- Često petlju koristimo kako bismo sekvencijalno pretražili neki izvor podataka. U takvim situacijama petlja gubi smisao u trenutku kada je podatak nađen

```
search = 5
for i in range(10):
    print("Current number: " , i)
    if(i==search):
        print("Number found!")
        break
```

Nakon ovog dela petlja nema smisao

- Nakon koda iz primera, znamo da je broj nađen, ali šta se dešava ako broj nije pronađen?

Slučaj kompletiranja petlje

- Petlje u Python-u imaju mogućnost obrade slučaja kompletiranja

  petlje komandom **else**

```
search = 15
for i in range(10):
    print("Current number: " , i)
    if(i==search):
        print("Number found!")
        break
else:
    print("Number not found")
```

```
Current number:  0
Current number:  1
Current number:  2
Current number:  3
Current number:  4
Current number:  5
Current number:  6
Current number:  7
Current number:  8
Current number:  9
Number not found
```

Vežba 8 (ppf-ex06 anim1.py)

- U programu je definisana promenljiva **target**
- Potrebno je animirati oznaku # tako da se kreće s leva na desn[redacted]# [redacted]e definisane promenljivom target

Napomena:

Moguće je vratiti se na početak reda u konzoli oznakom \r

```
print("\r",end="")
```

Moguće je pauzirati izvršavanje programa dodavanjem sledećih linija

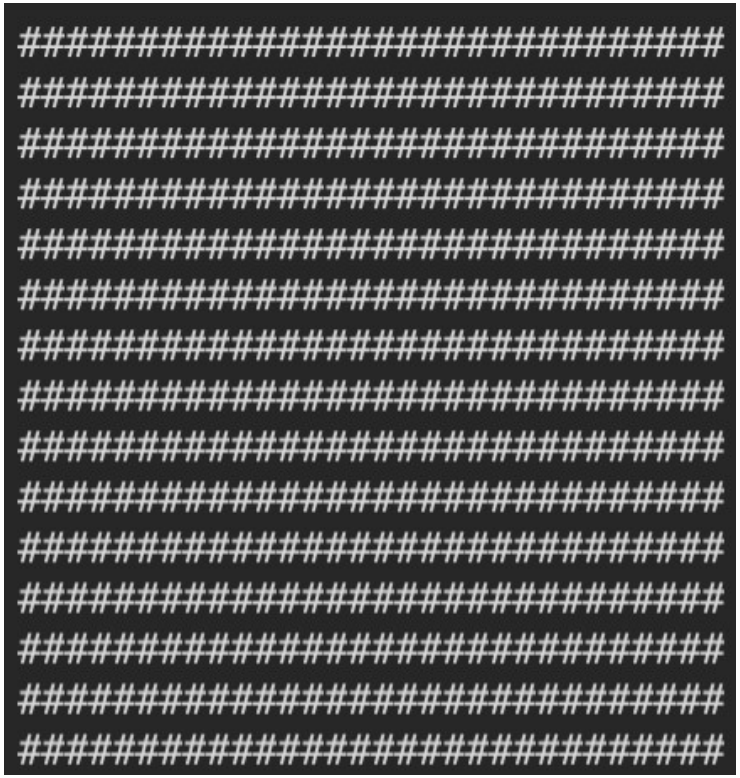
```
import time #na početku fajla
…
time.sleep(0.1) #tamo gde je potrebno pauzirati. 0.1 = 100 milisekundi
```

Vežba 9 (ppf-ex06 frame.py)

- U programu postoje promenljive w i h koje predstavljaju širinu i visinu matrice
- Potrebno je dopuniti program tako da se u konzoli prikaže matrica širine w i visine h
- Svaka pozicija u matrici treba da bude predstavljena znakom #
- Matrica treba da se briše i ponovo iscrtava u vremenskom intervalu

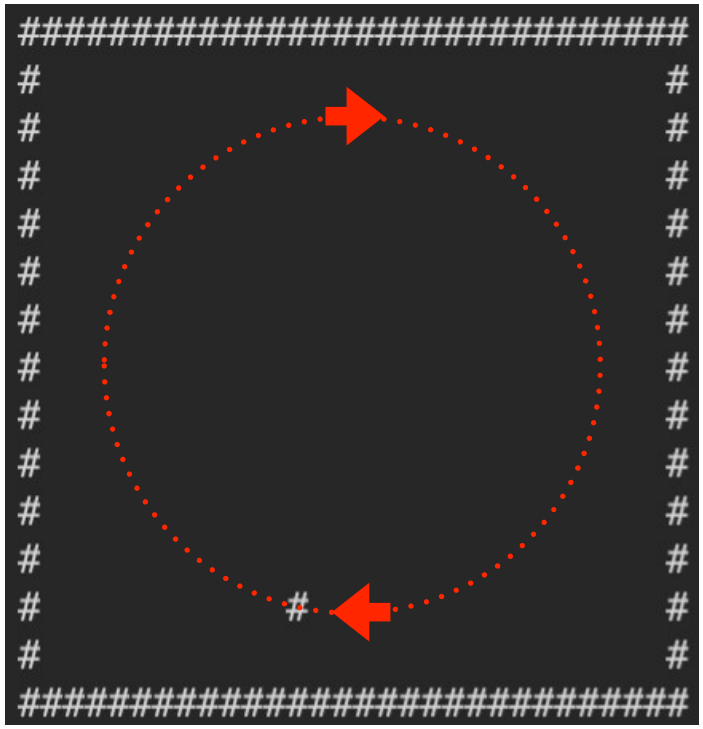
Napomena:
Za brisanje konzole dodati:

```
import os
...
os.system("clear")
```



Vežba 10 (ppf-ex06 rotation.py)

- Kreirati matricu određene veličine
- Uokviriti matricu oznakom # ili nekom drugom
- Unutar matrice, animirati oznaku # tako da se kreće kružno ili po nekom drugom šablonu



Napomena:
Sledeća forma izračunava rotaciju tačke:

```
import math
…
ptx1 = (math.cos(alpha) * ptx) + (-math.sin(alpha) * pty)
pty1 = ((math.sin(alpha)) * ptx) + (math.cos(alpha) * pty)
```